How to Find Unused Shared Libraries of a Process

Ravi Sankar Guntur
Contact: ravi.g@samsung.com
Samsung India

Contents

Who are we?

- Part of System Team working on LiMO Platform in Samsung India, Bangalore.
- Improving application launch times is one of our objective.

What is LiMO?

- Complete middleware and base application functionality
- Broad use of open source technologies

My Test System details

Processor: cortex-a8
RAM: 256 MB

Kernel: 2.6.32
gcc: 4.4.1

What constitutes application launch time?

Time taken from touch event upto displaying application's first GUI window is its launch time.

- Application launch time can be divided in to three phases
  - Kernel
  - Dynamic Linker
  - Application Initialization code.

What constitutes application launch time?

Time taken from touch event upto displaying application first GUI window is its launch time.

- Application launch time can be divided in to three phases
  - **Kernel**
    - **fork(), execve()**
    - **Setting up process image**
  - Dynamic Linker
    - Determine load dependencies
    - Relocate the application and all its dependencies
    - Initialize the application and dependencies in correct order
  - Application Initialization code.
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

What constitutes application launch time?

Time taken from touch event upto displaying application first GUI window is its launch time.

- Application launch time can be divided in to three phases
  - Kernel
    - fork(), execve()
    - Setting up process image
  - **Dynamic Linker**
    - **Determine load dependencies**
    - Relocate the application and all its dependencies
    - Initialize the application and dependencies in correct order
  - Application Initialization code.
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

Loading Dependencies

```
/mnt/nfs/gits/tools/trimlib # strace -T ./foo
execve("./foo", ["./foo"], [/* 38 vars */]) = 0 <0.001836>
brk(0)                                  = 0x11000 <0.000053>
uname({sys="Linux", node="H2_SDK", ...}) = 0 <0.000054>
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory) <0.000084>
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x4001d000 <0.000068>
open("/usr/lib/libsys-assert.so", O_RDONLY) = 3 <0.000086>
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0(\0\1\0\0\0\344\f\0\0004\0\0\0"..., 512) = 512 <0.000063>
fstat64(3, {st_mode=S_IFREG|0644, st_size=15876, ...}) = 0 <0.000056>
mmap2(NULL, 48364, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x40026000 <0.000064>
mprotect(0x4002a000, 28672, PROT_NONE)  = 0 <0.000073>
mmap2(0x40031000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3) = 0x40031000 <0.000077>
close(3)                                = 0 <0.000047>
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory) <0.000064>
open("/etc/ld.so.cache", O_RDONLY)      = 3 <0.000071>
fstat64(3, {st_mode=S_IFREG|0644, st_size=60027, ...}) = 0 <0.000055>
mmap2(NULL, 60027, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40032000 <0.000063>
close(3)                                = 0 <0.000046>
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory) <0.000065>
open("/usr/lib/libz.so.1", O_RDONLY)    = 3 <0.000078>
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0(\0\1\0\0\0\320\26\0\0004\0\0\0"..., 512) = 512 <0.000057>
fstat64(3, {st_mode=S_IFREG|0644, st_size=86936, ...}) = 0 <0.000049>
mmap2(NULL, 118288, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x40041000 <0.000064>
mprotect(0x40056000, 28672, PROT_NONE)  = 0 <0.000072>
mmap2(0x4005d000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14) = 0x4005d000 <0.000075>
close(3)                                = 0 <0.000046>
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory) <0.000063>
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3 <0.000072>
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0(\0\1\0\0\0\20,\0\0004\0\0\0"..., 512) = 512 <0.000056>
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x4001f000 <0.000056>
fstat64(3, {st_mode=S_IFREG|0644, st_size=42884, ...}) = 0 <0.000049>
mmap2(NULL, 74272, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x4005e000 <0.000064>
mprotect(0x40068000, 32768, PROT_NONE)  = 0 <0.000069>
mmap2(0x40070000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa) = 0x40070000 <0.000075>
close(3)                                = 0 <0.000047>
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory) <0.000063>
open("/lib/libc.so.6", O_RDONLY)        = 3 <0.000075>
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0(\0\1\0\0\0\234V\1\0004\0\0\0"..., 512) = 512 <0.000055>
fstat64(3, {st_mode=S_IFREG|0755, st_size=1189428, ...}) = 0 <0.000048>
mmap2(NULL, 1225988, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x40071000 <0.000068>
mprotect(0x4018f000, 32768, PROT_NONE)  = 0 <0.000072>
mmap2(0x40197000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11e) = 0x40197000 <0.000083>
mmap2(0x4019a000, 9476, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x4019a000 <0.000072>
```

Loading Dependencies.. (cont)

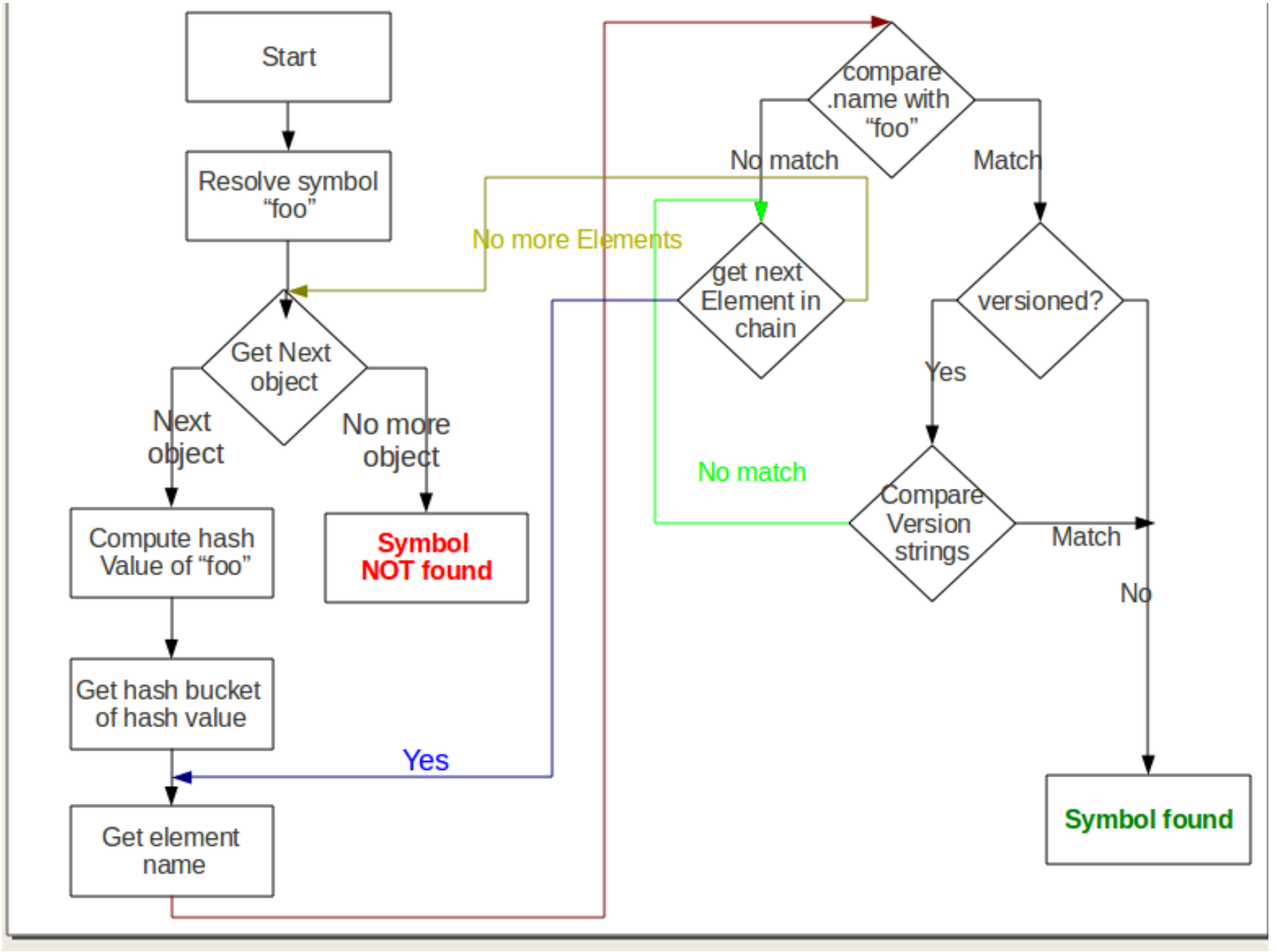
- Six to eight expensive calls like open(), read(), mmap(), stat(), close() per DSO
- Typical GUI applications have large set of DSO dependencies.
- *The more the number of participating Shared Libraries the more the time spent by dynamic linker to load those using open(), read(), mmap2(), close() calls.*

What constitutes application launch time?

Time taken from touch event upto displaying application first GUI window is its launch time.

- Application launch time can be divided in to three phases
  - Kernel
    - fork(), execve()
    - Setting up process image
  - **Dynamic Linker**
    - Determine load dependencies
    - **Relocate the application and all its dependencies**
    - Initialize the application and dependencies in correct order
  - Application Initialization code.
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

Relocation process



What constitues application launch times?

Time taken to display default GUI window for every application is its laucnh time.

- Application launch time can be divided in to three phases
  - Kernel
    - fork(), execve()
    - Setting up process image
  - **Dynamic Linker**
    - Determine load dependencies
    - Relocate the application and all its dependencies
    - **Initialize the application and dependencies in correct order**
  - Application Initialization code.
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

DSO constructors and destructors

Author of a DSO can write DSO initialization and deinitialization code.

- These init and deinit code can use symbols from other DSOs as well.
- Sorting of these init and deinit fucntions is a time consuming step.

What constitues application launch times?

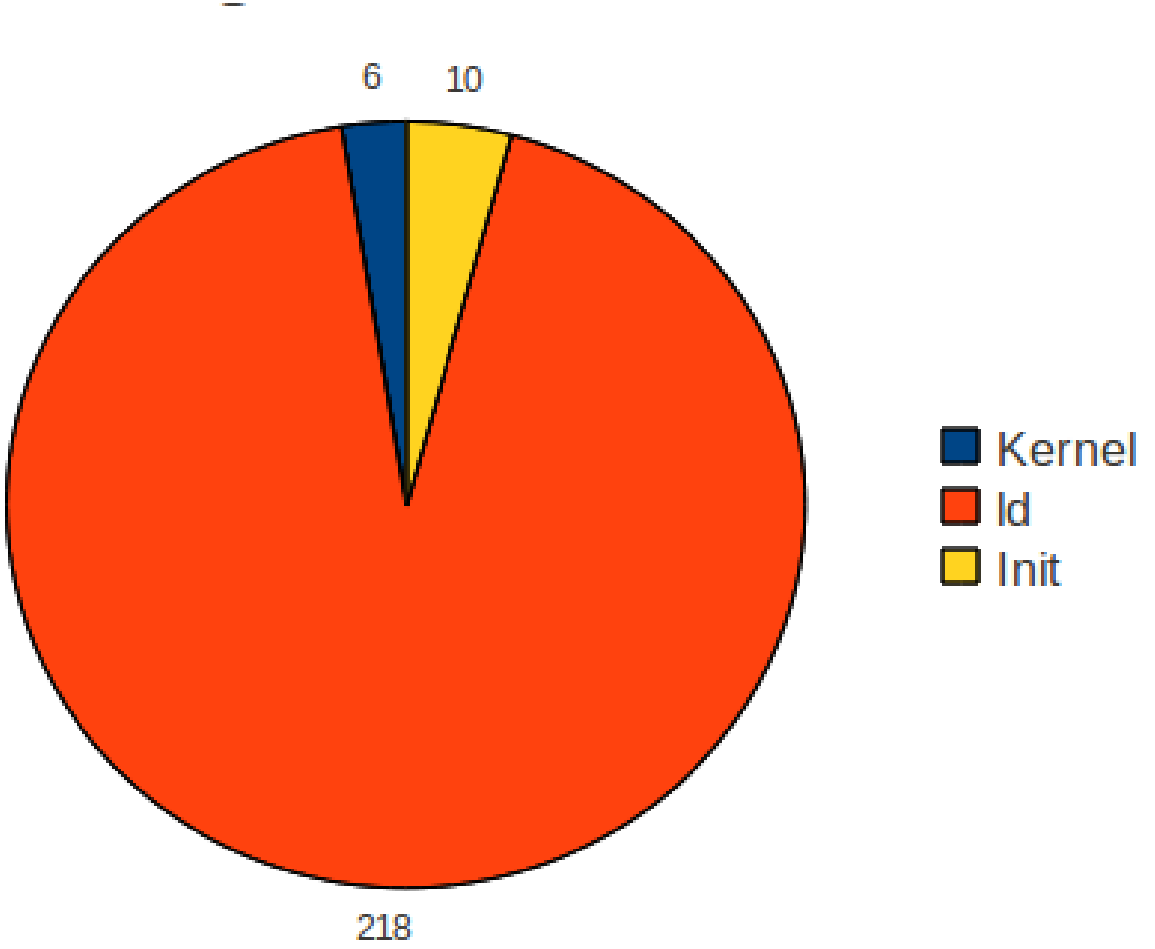
Time taken to display default GUI window for every application is its laucnh time.

- Application launch time can be divided in to three phases
  - Kernel
    - fork(), execve()
    - Setting up process image
  - Dynamic Linker
    - Determine load dependencies
    - Relocate the application and all its dependencies
    - Initialize the application and dependencies in correct order
  - **Application Initialization code.**
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

Application launch phases - summary

- Application launch time can be divided in to three phases
  - Kernel
    - fork(), execve()
    - Setting up process image
  - Dynamic Linker -> The more DSOs, the more higher startup time
    - Determine load dependencies
    - Relocate the application and all its dependencies
    - Initialize the application and dependencies in correct order
  - Application Initialization code. -> Mimimum required init code
    - Common stuff like, g_type_init(), dbus_init() etc
    - App specific init stuff like default GUI windows

Application launch phases – timing info (in ms)



Breakup of startup time for dialer application

Few known methods to improve app launch time

- Readahead: File pre-fetching method

- Prelinking: Program which modifies ELF files, so that the time which dynamic linker needs for their relocation at startup significantly decreases

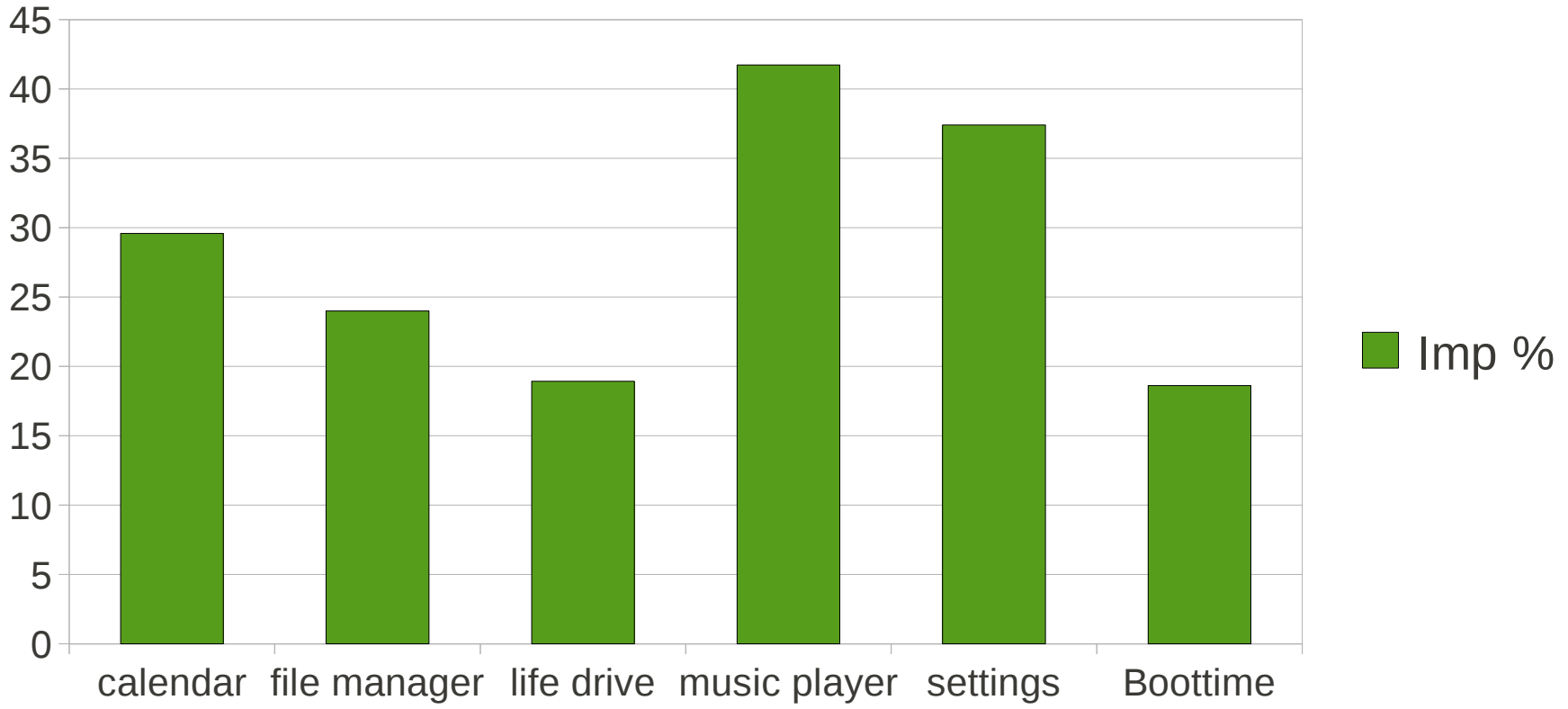
- Preloading: Reuse the relocation information for subsequent app launch.

- *All the above techniques assume that the application is linked with the right set of shared libraries and hence no unwanted libraries are loaded by them at runtime.*

- "ldd" gives unused direct dependencies of an ELF file.

- As-needed: flag to force the linker to link in the produced binary only the libraries containing symbols actually used by the binary itself.

Improvement in application launch times with “as-needed”



Auditing API for the GNU dynamic linker

- Auditing API allows an application to be notified when various dynamic linking events occur.
- Create a shared library that implements a standard set of function names.
- Set the environment variable LD_AUDIT to a list of shared libraries.

Auditing API for the GNU dynamic linker.. (cont)

*unsigned int la_objopen(struct link_map *map, Lmid_t lmid, uintptr_t *cookie);*

- **When a new shared object is loaded.**

*unsigned int la_objclose(uintptr_t *cookie);*

- After any finalization code for the object has been executed, before the object is unloaded.

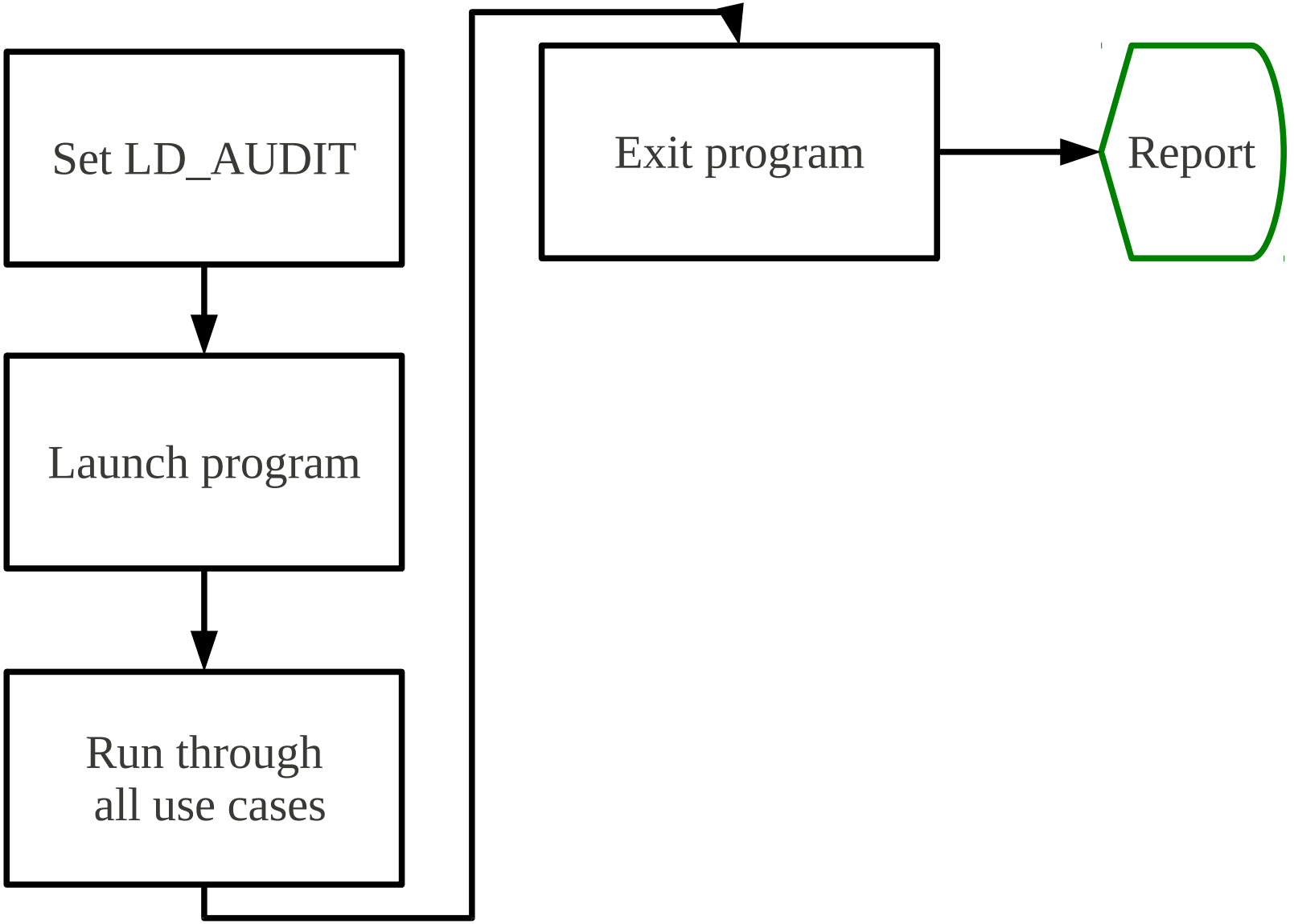
*uintptr_t la_symbind32(Elf32_Sym *sym, unsigned int ndx, uintptr_t *refcook, uintptr_t *defcook, unsigned int *flags, const char *symname);*

- When a symbol binding occurs

Audit Events

LD_AUDIT Event	Callback Action – Data update
objopen	Library loaded
objclose	Library unloaded
symbind	Library used
preinit	Get timing info (diff of time from constructor execution to preinit execution)

How to use libaudit.so



Sample report for “dialer” application

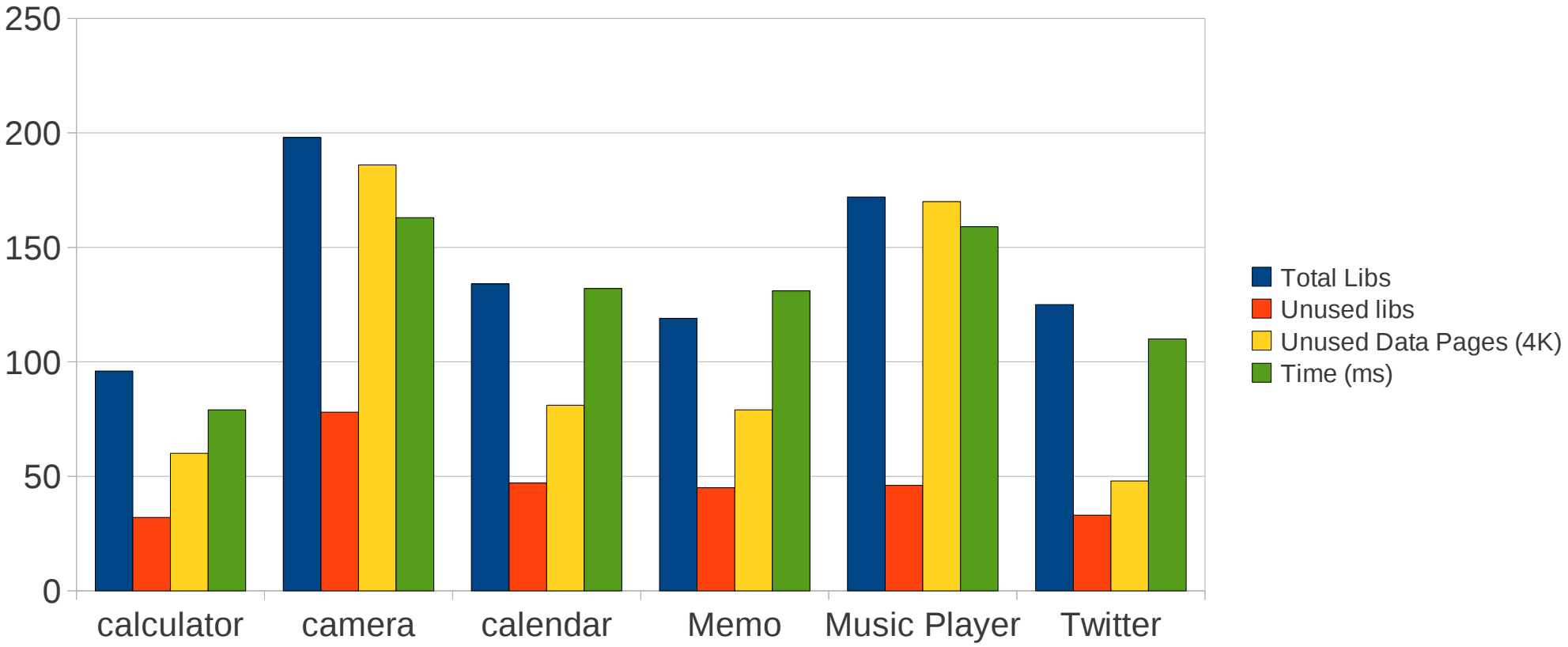
```
trimlib utility version 0.4
Program          PID     Used    RefCount        Data Pages      Library
dialer           2102    Yes     3               1               /lib/ld-linux.so.3
dialer           2102    No      0               1               /usr/lib/libsys-assert.so
dialer           2102    Yes     1               1               /usr/lib/libappcore-efl.so.1
dialer           2102    Yes     5               1               /usr/lib/libappcore-common.so.1
dialer           2102    Yes     5               1               /usr/lib/libaul.so.0
dialer           2102    Yes     18              1               /usr/lib/libdlog.so.0
dialer           2102    Yes     417             18              /usr/lib/libevas.so.1
dialer           2102    Yes     90              2               /usr/lib/libedje.so.1
dialer           2102    Yes     466             3               /usr/lib/libelementary-ver-pre-svn-07.so.0
dialer           2102    Yes     21              1               /usr/lib/libui-gadget.so.0.1.0
dialer           2102    Yes     30              1               /usr/lib/libbundle.so.0
dialer           2102    Yes     20              1               /usr/lib/libvconf.so.0
dialer           2102    Yes     138             3               /lib/libpthread.so.0
dialer           2102    Yes     841             5               /lib/libc.so.6
dialer           2102    Yes     96              15              /usr/lib/libecore.so.1
dialer           2102    Yes     66              2               /usr/lib/libecore_x.so.1
dialer           2102    Yes     3               1               /usr/lib/libecore_input.so.1
dialer           2102    Yes     15              1               /lib/libdl.so.2
dialer           2102    Yes     3               1               /usr/lib/libsysman.so.0
dialer           2102    Yes     36              1               /usr/lib/libgobject-2.0.so.0
dialer           2102    No      0               3               /usr/lib/libsensor.so
dialer           2102    Yes     136             1               /usr/lib/libglib-2.0.so.0
dialer           2102    Yes     3               1               /usr/lib/librua.so.0
dialer           2102    Yes     193             4               /usr/lib/libX11.so.6
dialer           2102    Yes     1               1               /usr/lib/libscreen-engine.so.0
dialer           2102    Yes     19              1               /usr/lib/libdbus-glib-1.so.2
dialer           2102    Yes     145             1               /usr/lib/libdbus-1.so.3
dialer           2102    Yes     72              2               /usr/lib/libsqlite3.so.0
dialer           2102    No      0               1               /usr/lib/libxdgmime.so.1
dialer           2102    Yes     298             1               /usr/lib/libeina.so.1
```

```
Library coverage summary for dialer: 2102
Total libraries 119, Unused 28, Unsued % 23.529411
Time spent in ld from AUDIT init to main() 91 milliseconds
Total Data Segment size of unsued Libraries is 60 Pages of each 4K (Total 245760 Bytes)
```

Known issues with LD_AUDIT

- LD_AUDIT does not work with Shared Libraries with no code in them.
- Example ICU-4.0 “libicudata.so”
- Error: “no PLTREL found in object /usr/lib/libicudata.so.40”
- Recompile after patching libicudata by sed'ing -nostdlib etc away sed -i --"s/-nodefaultlibs -nostdlib//" config/mh-linux

Library coverage summary for few apps

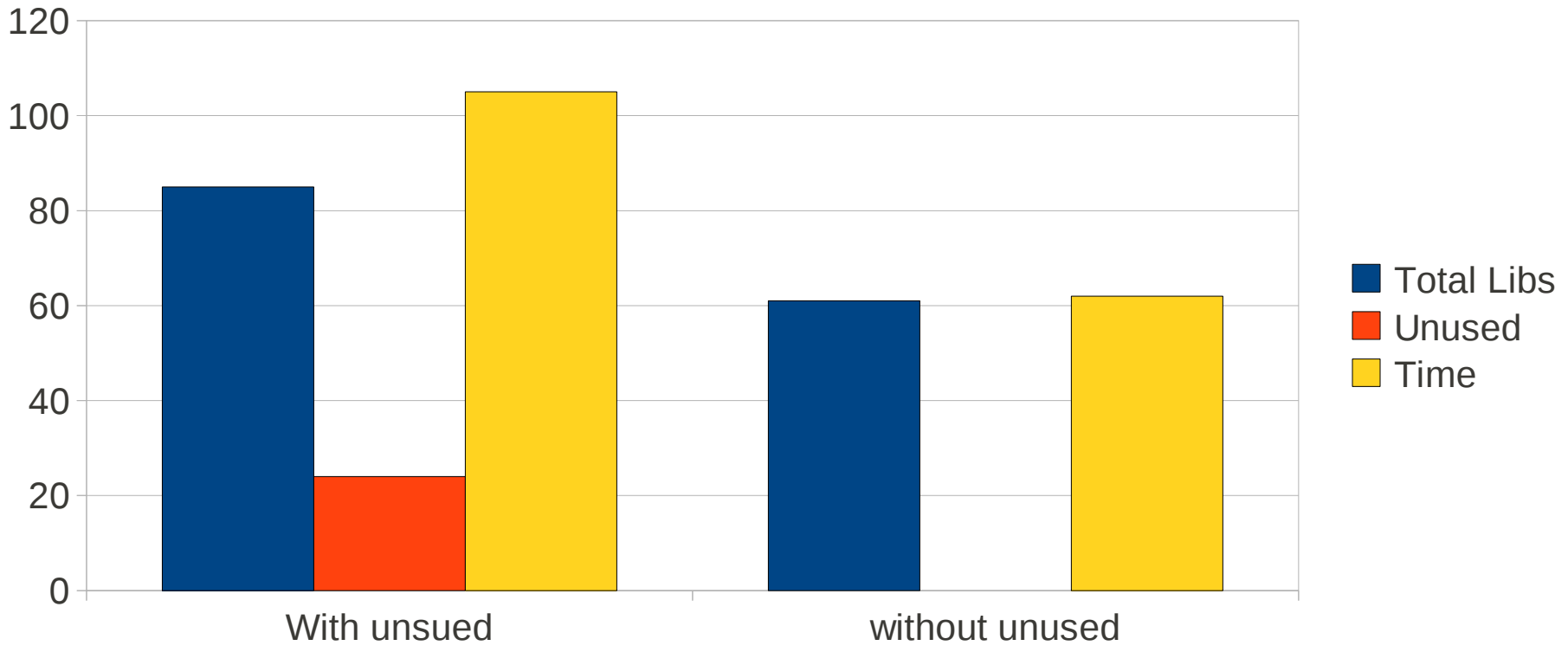


Findings of libaudit.so utility for few sample set of applications

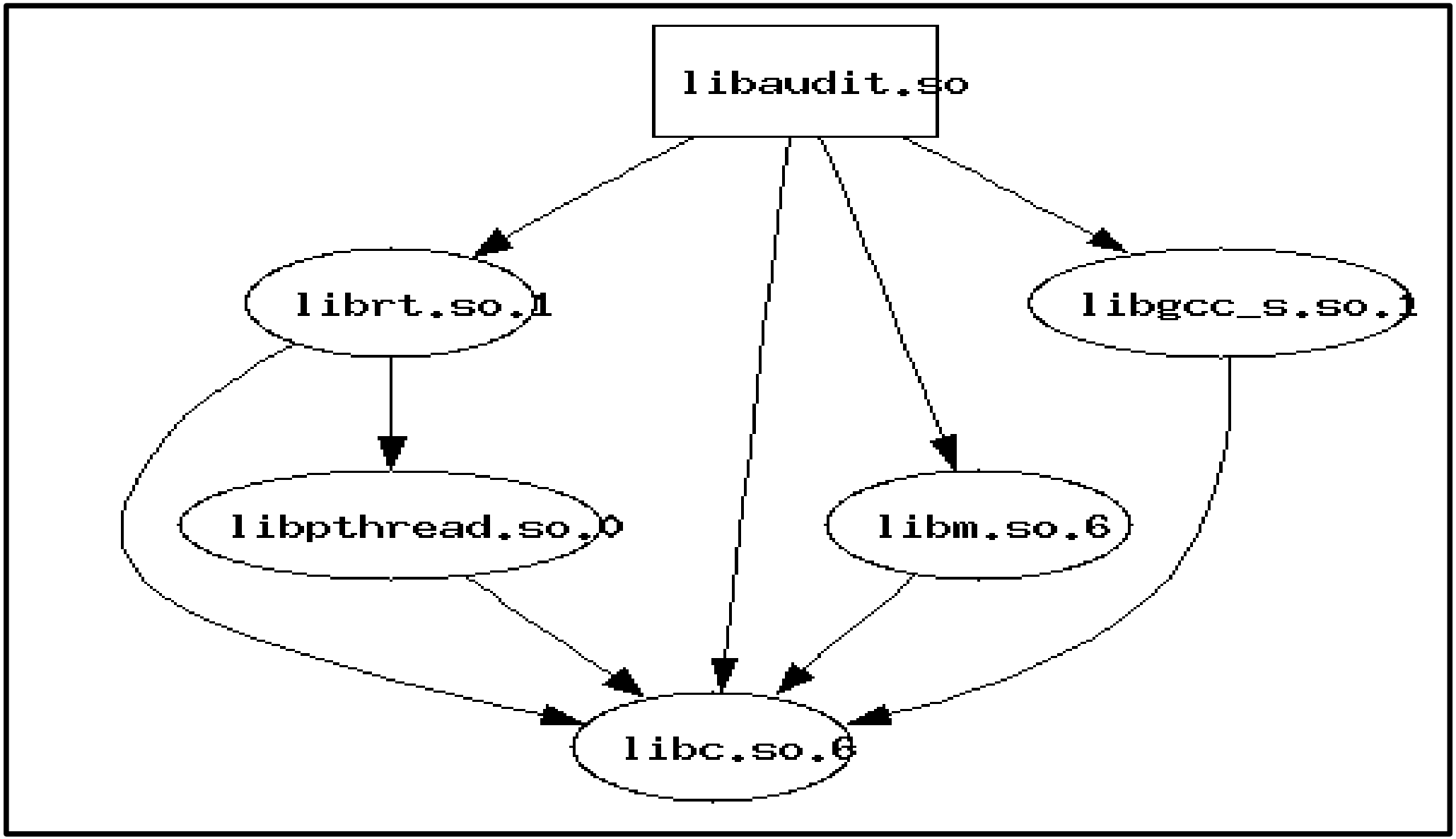
Summary of the findings

- ✔ Average percentage of unused libraries is around 25%
- ✔ Memory loaded due to data segment of unsued library is unused

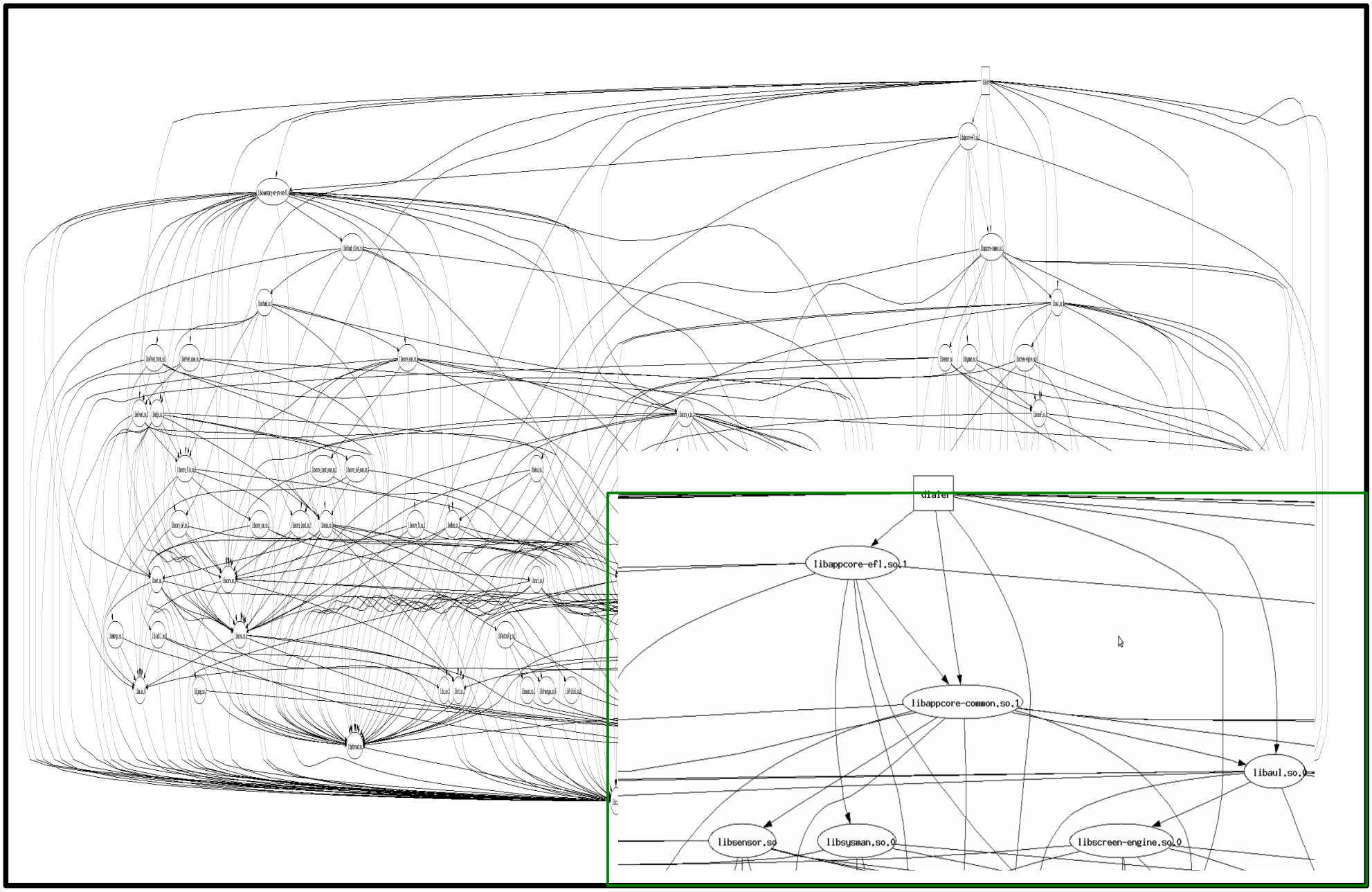
Simulation test results



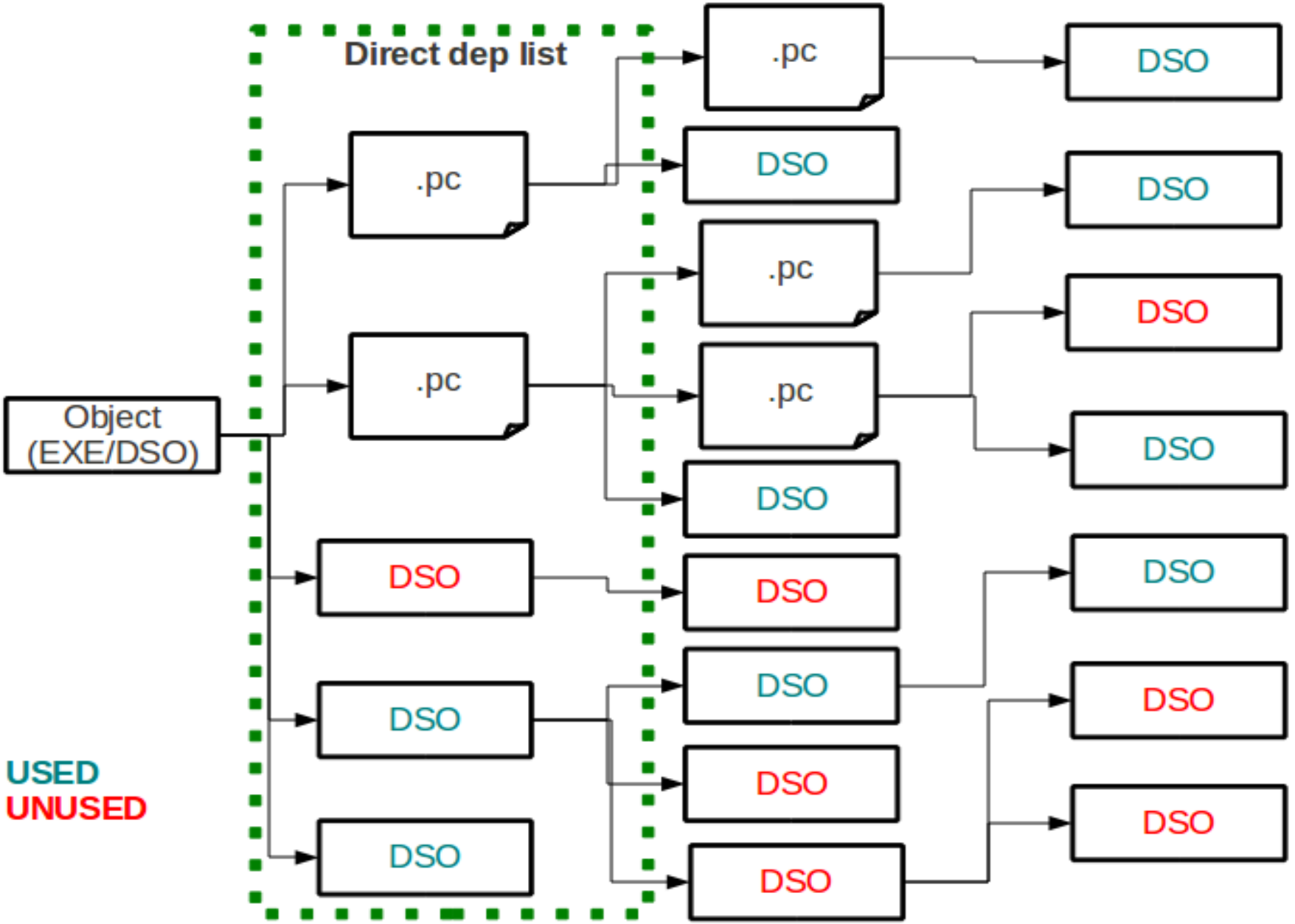
Dependency graph of libtrim.so.0.5



Dependency graph of “dialer” app



Source of unused DSOs



References

- ELF standard - http://refspecs.freestandards.org/elf/elf.pdf
- How to Write Shared Libraries by Ulrich Drepper - http://www.akkadia.org/drepper/dsohowto.pdf
- Libelf - http://www.mr511.de/software/english.html
- How to use GNU Linker flag “--as-needed”- http://www.gentoo.org/proj/en/qa/asneeded.xml
- Plot dependency graph of a ELF file - http://bit.ly/grMehw
- Computer Science from the Bottom Up - http://bottomupcs.sourceforge.net/csbu/book1.htm
- Graphviz documentaiton - http://www.graphviz.org/
- Manual page for “rt-ld-audit”
- Manual page for “ld.so”
- Manual page for “ldd”
- LiMO foundation web site - http://www.limofoundation.org

The End

Backup slides

Library usage statistics in SLP

```
/mnt/nfs/gits/tools/trimlib # ./stats.sh
161: 2098: /opt/apps/deb.com.samsung.email/bin/email
156: 14986: /opt/apps/deb.com.samsung.browser/bin/browser
138: 2035: /usr/bin/wrt_engine_daemon
135: 2110: /opt/apps/deb.com.samsung.music-player/bin/music-player
122: 14976: /opt/apps/deb.com.samsung.message/bin/message
121: 1813: /usr/bin/menu_screen__AUL_STARTTIME__1301290153/977877__AUL_CALLER_PID__1788
120: 2048: /usr/bin/java-push-server
119: 2118: /opt/apps/deb.com.samsung.gallery/bin/gallery-beat
103: 2139: /opt/apps/deb.com.samsung.contacts/bin/contacts
102: 2128: /opt/apps/deb.com.samsung.myfile/bin/myfile
102: 1783: /usr/bin/mtp-uikies
99: 1711: /usr/bin/usb_setting
95: 2022: /usr/lib/scim-1.0/scim-helper-launcher--daemon--configsocket--display:0ise-moakey-koreanb70bf6cc-ff77-47dc-a137-60acc32d1e0c
94: 1984: /usr/bin/cbhm
93: 1757: /opt/apps/deb.com.samsung.indicator/bin/indicator
85: 1598: /usr/bin/launchpad_preloading_preinitializing_daemon
82: 1683: /usr/bin/enlightenment-i-really-know-what-i-am-doing-and-accept-full-responsibility-for-it
81: 1788: /usr/bin/menu_daemon
79: 1907: /usr/bin/isf-panel-efl
53: 1944: /usr/bin/msg-server
52: 1619: /usr/bin/system_server
45: 1973: /usr/bin/lbs_server
44: 1682: /usr/lib/sapwood/sapwood-server
43: 1942: /usr/bin/email-service
42: 1918: /usr/bin/file-manager-server
38: 1603: /usr/bin/pulseaudio--log-level=4--system-D
35: 1916: /usr/bin/dnet
35: 1655: /usr/bin/sound_server-S
34: 2013: /usr/bin/power_manager-d-x
33: 1649: /usr/bin/sf_server-s/usr/etc/sf_sensor.conf-f/usr/etc/sf_filter.conf-p/usr/etc/sf_processor.conf-d/usr/etc/sf_data_stream.conf
32: 1581: /usr/bin/Xorg:0-logfile/opt/var/log/Xorg.0.log-ac-noreset-r+accessx0-config/opt/etc/X11/xorg.conf-configdir/opt/etc/X11/xorg.conf.d
28: 1909: /usr/bin/contacts-svc-helper
27: 1594: /usr/bin/telephony-server
26: 1960: /usr/bin/brcm_gps_daemon/opt/data/lbs-engine/brcm/brcm_gps_config.xml
26: 1948: /usr/bin/alarm-server
26: 1576: /usr/bin/audio-session-mgr-server
25: 1950: /usr/bin/bluetooth-agent
```

Dependency graph of “dialer app”

```
sbox-slp2.0-arm: /host/gits/tools/trimlib] > ./dependencies.sh /opt/apps/deb.com.samsung.dialer/bin/dialer dialer.png
Analyzing dependencies of: /opt/apps/deb.com.samsung.dialer/bin/dialer
Creating output as:        dialer.png

/opt/apps/deb.com.samsung.dialer/bin/dialer ===> libappcore-efl.so.1
/usr/lib/libappcore-efl.so.1 ===> libelementary-ver-pre-svn-07.so.0
/usr/lib/libelementary-ver-pre-svn-07.so.0 ===> libehal.so.1
/usr/lib/libehal.so.1 ===> libedbus.so.1
/usr/lib/libedbus.so.1 ===> libecore.so.1
/usr/lib/libecore.so.1 ===> libeina.so.1
/usr/lib/libeina.so.1 ===> librt.so.1
/lib/librt.so.1 ===> libc.so.6
/lib/libc.so.6 ===> ld-linux.so.3
/lib/librt.so.1 ===> libpthread.so.0
/lib/libpthread.so.0 ===> libc.so.6
/lib/libpthread.so.0 ===> ld-linux.so.3
/lib/librt.so.1 ===> ld-linux.so.3
/usr/lib/libeina.so.1 ===> libm.so.6
/lib/libm.so.6 ===> ld-linux.so.3
/lib/libm.so.6 ===> libc.so.6
/usr/lib/libeina.so.1 ===> libdl.so.2
/lib/libdl.so.2 ===> libc.so.6
/lib/libdl.so.2 ===> ld-linux.so.3
/usr/lib/libeina.so.1 ===> libgcc_s.so.1
/lib/libgcc_s.so.1 ===> libc.so.6
/usr/lib/libeina.so.1 ===> libpthread.so.0
/usr/lib/libeina.so.1 ===> libc.so.6
/usr/lib/libecore.so.1 ===> libglib-2.0.so.0
/usr/lib/libglib-2.0.so.0 ===> libgcc_s.so.1
/usr/lib/libglib-2.0.so.0 ===> libc.so.6
/usr/lib/libecore.so.1 ===> librt.so.1
/usr/lib/libecore.so.1 ===> libm.so.6
/usr/lib/libecore.so.1 ===> libgcc_s.so.1
/usr/lib/libecore.so.1 ===> libpthread.so.0
/usr/lib/libecore.so.1 ===> libc.so.6
```

Who should be concerned?

- Embedded Developers who run their applications on Low-end processors and have limited main memory

- When you are told to optimize the application launch time. And you notice that most of the starup time is spend even before your “main” is called.

- As a System Engineer I am interested in each Process “**Library Covergae**”.

What are Shared Object Dependencies?

$ gcc -g foo.c /usr/lib/libz.a -o foo

- Link editor extracts archive library members and copies them into the output object file.

- These statically linked services are available during execution without involving the dynamic linker.

- Executables or Shared library can depend on another Shared Library for services.

$ gcc -g foo.c -lz -o foo

- The Link editor inserts supplied Shared library names as “NEEDED” list in to the Object File.

- The dynamic linker loads the “NEEDED” shared object files to the process image for execution.

- Thus ELF standard defines a way for executable and shared object files to describe their specific dependencies in section called “dynamic section”.

What is Direct and Indirect dependency?

- When a Shared Object or Executbale is linked against another Shared Object then we call it as “Direct dependency”

- Entry is made for each dependent Shared Object in the Object's Dynamic Section itself.

- Each Shared Object mentioned in the Dynamic Section of a given Object can further bring its own set of Dependency list. In this case we call it as indirect dependency list.

- $gcc -g -shared -fPIC foo.c -o libfoo.so -lz
- $gcc -g bar.c -o bar -lfoo

- foo is direclty dependent on libbar.so and indirectly depends on libz.so.

How to find Direct dependency?

$ readelf -d <ELF File>

```
Dynamic section at offset 0xf20 contains 21 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x80482bc
 0x0000000d (FINI)                       0x804849c
 0x00000004 (HASH)                       0x804818c
```

$ gcc -g foo.c /usr/lib/libz.a -o foo

```
Dynamic section at offset 0xf18 contains 22 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libz.so.1]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x80482f0
 0x0000000d (FINI)                       0x80484cc
 0x00000004 (HASH)                       0x804818c
```

Entry is made for libz.so

$ gcc -g foo.c -lz -o foo

How to find Direct & Indirect dependency?

$ ldd <ELF file>

```
ravi@ravi-desktop:~$ readelf -d   /usr/lib/libdbus-glib-1.so.2.1.0

Dynamic section at offset 0x1be68 contains 28 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libdbus-1.so.3]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libgobject-2.0.so.0]
 0x00000001 (NEEDED)                     Shared library: [libgthread-2.0.so.0]
 0x00000001 (NEEDED)                     Shared library: [librt.so.1]
 0x00000001 (NEEDED)                     Shared library: [libglib-2.0.so.0]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000e (SONAME)                     Library soname: [libdbus-glib-1.so.2]
 0x0000000c (INIT)                       0x5a80
```

```
ravi@ravi-desktop:~$ ldd  /usr/lib/libdbus-glib-1.so.2.1.0
        libdbus-1.so.3 => /lib/libdbus-1.so.3 (0x00a40000)
        libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0x00f72000)
        libgobject-2.0.so.0 => /usr/lib/libgobject-2.0.so.0 (0x00206000)
        libgthread-2.0.so.0 => /usr/lib/libgthread-2.0.so.0 (0x00110000)
        librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0x00fb0000)
        libglib-2.0.so.0 => /lib/libglib-2.0.so.0 (0x00a88000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00245000)
        /lib/ld-linux.so.2 (0x00e90000)
        libpcre.so.3 => /lib/libpcre.so.3 (0x00bbd000)
ravi@ravi-desktop:~$
```

Brought in by libgobject-2.0.so.0

How Dependency information is supplied?

- Typical GUI applications have large set of shared object dependencies.
- pkg-config is a helper tool used when compiling applications and libraries.
- A package developer exports correct compiler options in a ".pc" file so that user can use it to expand to proper options rather than hard-coding.
- For instance glib exports compiler and linker options in its package config file located at "/usr/lib/pkgconfig/glib-2.0.pc"
- $ gcc `pkg-config --cflags --libs glib-2.0` gtk.c -fPIC -o gtk

```
ravi@ravi-desktop:~/sbox/gits/tools/trimlib$ cat /usr/lib/pkgconfig/glib-2.0.pc
prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include

glib_genmarshal=glib-genmarshal
gobject_query=gobject-query
glib_mkenums=glib-mkenums

Name: GLib
Description: C Utility Library
Version: 2.24.1
Libs: -L${libdir} -lglib-2.0
Libs.private:
Cflags: -I${includedir}/glib-2.0 -I${libdir}/glib-2.0/include
ravi@ravi-desktop:~/sbox/gits/tools/trimlib$ pkg-config --cflags --libs /usr/lib/pkgconfig/glib-2.0.pc
-I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include  -lglib-2.0
```

Summary of Relocation costs

✔ Performance of each lookup depends ~ on the length of the hash chains and the number of objects in the lookup scope.

✔ Length of Hash chains depends on number of Symbols exported – traddeoff between Speed and hash table memory overhead.

✔ Relocation is at least O(rs); where r is number of relocations and s is the number of symbols defined in all objects.

✔ The more DSOs participate or the more symbols are defined in the DSOs, the longer the symbol lookup takes.

✔ Reducing the number of loaded objects increases performance.

GNU Link editor option “--as-needed”

- link only the libraries containing symbols actually used by the binary itself.
- LDFLAGS="-Wl,--as-needed"
- Incorrect way of using “-as-needed”
- $ gcc -Wl,--as-needed -lm someunit1.o someunit2.o -o program
- in this case libm is considered before the object files and discarded independently from the content of the two.
- Correct way of using “-as-needed”
- $ gcc -Wl,--as-needed someunit1.o someunit2.o -lm -o program
- **“--as-needed” can not solve the problem introduced by bad library design choice.**

Finding unused direct dependencies with 'ldd'

$ ldd -u <ELF File>

- Will give list of direct unused dependencies of the ELF file.
- Checks for dependency with in a “ELF File context” but not a Process context.

ELF 1 --> ELF 2 --> ELF 4

|

--> ELF 5

--> ELF 3

- Increased usage of “dlopen” in service plugins. “ldd “ does not support dlopened libraries.

Source code for the libaudit.so tool to find unsed DSOs

git: git://gitorious.org/slp-siso/trimlib.git
GNU GPL license.